区块链 P2P 网络协议演进过程 *

武　岳，李军祥†

(上海理工大学 管理学院，上海 200093)

摘　要：区块链是一种分布式系统，使用点对点（peer to peer，P2P）网络作为区块链网络层通信协议，点对点的传播机制和验证机制共同构成了区块链网络层的基石。点对点网络所有节点共同承担点对点网络服务，弱化甚至没有中心服务器，其独有的特性使得区块链系统可以脱离中心服务器的束缚，做到真正的分布式、去中心化。针对点对点网络协议的这一特点，通过阅读源代码和官方文档，对比特币(Bitcoin)、以太坊(Ethereum)以及超级账本（hyperledger fabric）三种主流区块链系统的 P2P 协议进行了详细研究。通过讨论区块链点对点协议在演进过程中的变化分析了不同协议的优缺点，同时提出了分析标准，作出了量化评价。该比较以期为未来区块链网络协议的研究提供有益的启发与指导。

关键词：区块链；P2P 网络；比特币；以太坊；超级账本；Gossip；Kadenlia

中图分类号：TP312　　doi: 10.3969/j.issn.1001-3695.2018.07.0365

Evolution process of blockchain P2P network protocol

Wu Yue, Li Junxiang†

(*Business School*, *University of Shanghai for Science & Technology*, *Shanghai* 200093, *China*)

Abstract: Blockchain is a distributed system, which uses peer to peer (P2P) network as the communication protocol of the blockchain network layer. The communication and verification mechanisms of P2P jointly constitute the cornerstone of the blockchain network layer. All nodes of P2P network share P2P network services, weakening or even without central server, and its unique features enable the blockchain system to break away from the central server to achieve truly distribution and decentralization. By the reading of the source code and the official documents, the paper analyzed P2P protocols of three main blockchain systems, Bitcoin, Ethereum and hyperledger fabric, in detail, discussed the P2P protocol of block chain in the change of the evolution process and analyzed the advantages and disadvantages of different protocols. The comparison puts forward an analysis standard and a quantitative evaluation. The comparison hopes to offer a useful inspiration and guidance for the research of blockchain network protocols in future.

Key words: blockchain; P2P network; Bitcoin; Ethereum; hyperledger fabric; Gossip; Kadenlia

0　引言

区块链是以加密机制、储存机制、共识机制等多种技术组成的分布式系统，可以在无中心服务器的情况下实现相互信任的点对点交易功能。区块链最大的特点是去中心化和分布式，参与节点间通过区块链共识机制使得参与节点共同提供系统服务，实现中心化系统如金融中介机构的功能。共识机制是网络大部分节点在约定时间段内对交易状态达成相同的确认，其中共识性通过工作量证明(POW)、权益证明(POS)、实用拜占庭容错(PBFT)等多种共识算法实现。而一致性则要求大部分节点在约定时间段内保持数据内容一致，从而利用共识算法实现共识。传统中心化网络系统需要中心提供强大稳定的服务器和足够的带宽来支持客户端节点使用。通过 P2P 网络，区块链系统可以在没有中心服务器的情况下达到快速同步数据，实现共识机制的一致性。如图 1 所示，共识算法和 P2P 网络分别承担共识性和一致性的任务，共同组成区块链的共识机制，从而实现区块链的分布式和去中心化。

罗杰文[1]根据是否中心化及结构化提出 P2P 网络分类标准。周文莉等人[2]提出了 P2P 网络的五种结构分类。王学龙等人[3]根据 P2P 关键技术类型，对 P2P 网络分类模型进行讨论。刘小敏等人[4]对 P2P 网络检索机制进行了详细描述。 谭庆丰等人[5]提出了基于 P2P 的隐秘通信方式。邵奇峰和刘敖迪等人[6,7]在区块链体系中介绍了区块链网络层。高峰等人[8]提出邻居节点识别溯源模型。叶聪聪等人[9]提出了评估和检查区块链安全性的

收稿日期：2018-07-13；**修回日期：**2018-08-30　**基金项目：**国家自然科学基金资助项目（71572113，71432007）；国家自然科学基金匹配项目（IP16303003，2017KJFZ024，2018KJFZ035，CFTD17004Z）

作者简介：武岳（1988-），男，江苏南京人，博士研究生，主要研究方向为区块链、分布式系统、智能合约；李军祥（1971-），男（通信作者），山东乐陵人，副教授，博导，主要研究方向为管理科学与工程、系统科学（lijx@usst.edu.cn）.

模型。Fukumits 等人[10]提出基于区块链的保密共享 P2P 储存协议。Gattermayer 等人[11]提出了基于区块链的 P2P 集群多级评分系统。区块链产品由于其财富创造效应，吸引大量资金的投入，使得区块链产品和技术快速创新演进。然而相关学术研究严重滞后，亟待跟进。截止 2018 年 7 月，通过知网、万方搜索区块链、P2P 网络并无相关中文文献，以谷歌学术搜索 blockchain、P2Pnetwork，仅有比特币[12]相关的 5 篇文献。

本文按区块链产品产生时间顺序，选择比特币、以太坊[13,14]、超级账本[15]三种作为研究对象。其中比特币和以太坊是最具影响力的分布式交易系统，其代币价值及整体市值稳居虚拟货币第一和第二位置；超级账本是最具影响力的企业级区块链产品，被广泛应用于各种领域。通过阅读区块链产品白皮书和源代码，系统性的分析多种区块链产品的 P2P 网络技术原理、核心优势、存在问题等内容，详细阐述区块链 P2P 网络演进过程中的变化过程和技术创新。

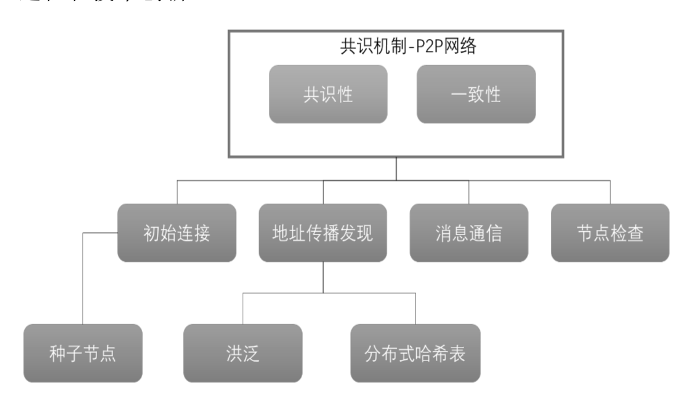


图 1　点对点网络在区块链中架构

1　P2P 网络与区块链

1.1　P2P 在区块链的应用

P2P 网络自身有多方面优点，在区块链的应用如下：

a)去中心化。区块链的资源和服务分布在所有参与节点上，通过共识机制维护区块链网络一致性，无须中心系统的存在。

b)可扩展性。区块链节点可以自由加入、退出，网络系统根据节点自由扩展。

c)健壮性。区块链网络没有中心节点，也就没有了攻击对象。参与节点分布在网络中，部分节点遭到破坏对区块链系统无影响。

d)高性价比。无须昂贵的专业设备搭建中央服务器，无须支付昂贵的宽带费用，普通用户也可以参与区块链。

e)隐私保护。区块信息采用广播机制，无法定位广播初始节点，防止用户通信被监听，保护用户隐私。

f)负载均衡。区块链通过限制节点连接数等配置，避免资源负载、网络阻塞。

1.2　区块链 P2P 网络分类

针对区块链应用特点，按照 P2P 网络是否去中心化、节点地址是否结构化两个方面，将 P2P 网络分为如下四类。网络架构如图 2 所示。

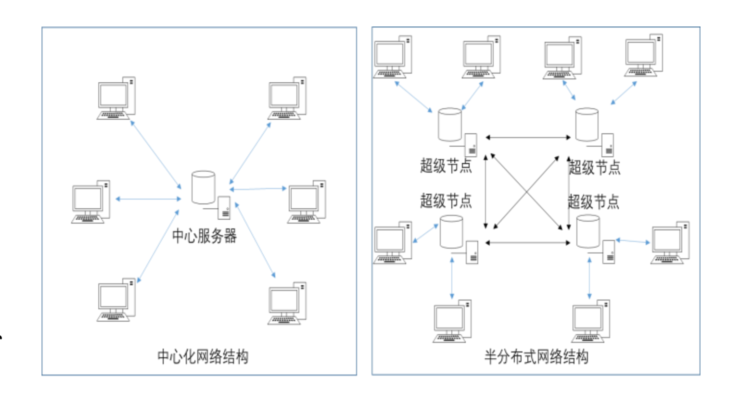


图 2　点对点网络结构

1）中心化 P2P 网络

中心服务器对于中心化网络结构至关重要，保存接入节点的地址信息是网络的核心。普通节点通过接入中心服务器获取其他节点地址，从而实现节点与节点之间的通信。著名的 MP3 共享软件 Napster[16]采用的就是中心化网络结构。将音乐文件与保存文件的节点相互关联，用户查找某个音乐时，中心服务器告知储存节点地址，用户点对点连接以获得音乐。中心服务器也决定着网络整体稳定性，一旦中心服务器出现问题，整个 P2P 网络就会瘫痪。这与传统中心化网络类似，如银行等金融机构；不同的是 P2P 中心服务器只提供索引，而传统中心服务器提供完全服务。P2P 网络节点可以直接与另外一个节点通信，而金融机构不允许节点间通信。

2）全分布式非结构化 P2P 网络

全分布 P2P 节点可以自由加入退出，并且没有中心节点，节点地址没有结构化统一标准，整个网络结构呈随机图的结构，无固定网络结构图。其成功的应用产品是 Gnutella[17]通信协议，它是一种点对点的搜索系统。Gnutella 采用的是洪泛(flooding)技术实现发现其他节点和随机转发的机制，采用 TTL (time to live)的减值来实现控制消息通信有限次数传播。然而完全的自由意味着新节点无法得知 P2P 网络节点信息，从而无法加入网络。Gnutella 也使用了目录服务器来方便节点得到其他节点的网络地址。全分布式 P2P 网络更加自由化的同时也带来节点管理的问题，节点频繁加入、退出使得整个网络结构无法稳定，大量的广播消息不仅造成资源浪费，甚至会阻塞网络。中本聪设计的比特币采用的就是这种 P2P 网络结构，全分布式使得任何人任何节点都可以参与，非结构化使得节点间既可以通过区块链 P2P 协议同步区块数据，又保持匿名隐私保护，使得区块链分布式去中心化概念深入人心。

3）全分布式结构化 P2P 网络

全分布式最大的问题在于节点地址管理，节点间没有固定规则约束，无法精确定位节点信息，只能通过洪泛查询方式进行查找，对网络的消耗很大。结构化网络采用分布式哈希表（distributed Hash table, DHT）[18]，通过如 Hash 函数一类的加密散列函数，将不同节点地址规范为标准长度数据。比较成功的案例有 Chord[19]、Pastry[20]等。网络结构同非结构化相同，都是随机形式，无固定结构，但节点管理有固定结构图。以太

坊将节点椭圆加密算法的公钥转换为 64 Byte 长度的 NodeID 作为唯一标志符来区分节点，使得以太坊可以在没有中心服务器的情况下实现节点地址精确查找。

4）半分布式 P2P 网络

结合中心化和分布式模型各有的优点，将节点分类成普通节点和超级节点，从而构成了半分布式网络结构。每个超级节点维护部分网络节点地址、文件索引等工作，超级节点共同实现中心服务器功能。超级节点本身却是分布式，可以自由扩展退出，具备分布式网络优点。Kazza[21]是其代表的成功的应用。超级账本 hyperledger fabric 采用的 P2P 网络结构就是半分布式，将节点分为普通用户节点和超级节点（排序、背书节点等）。超级节点可以由普通节点选举，也可以自行配置，单独一个超级节点停机不影响系统运行。超级账本所有交易必须通过超级节点认证，区块也是由超级节点生成，普通节点间可以互相同步区块。

不同 P2P 网络结构，各有自身的优缺点，也适应于不同的应用场景。比特币最初的目标是建立去中心化的交易平台，致力于代替银行等金融机构实现价值转移交换。同时，中心化 P2P 网络不符合区块链去中心化特征，没有相关产品，而中心化 P2P 网络类似传统的中介式机构如银行等，故本文所讨论的中心化系统指银行类金融中介机构，不具备 P2P 功能仅作为对标标的，以实现对区块链产品的对比评判。中心化 P2P 网络类似传统的中介式机构如银行等，是区块链产品致力于替代的标志系统，本文所讨论中心化系统，从 P2P 应用、区块链应用、是否去中心化、节点是否可精确查找等方面比较了四种 P2P 网络（表 1）。

表 1　点对点网络功能及特点对比

功能/特点	中心化点对点网络	全分布式非结构化点对点网络	全分布式结构化点对点网络	半分布式点对点网络
P2P 应用	纳普斯特(Napster)	努特拉(Gnutella)	科德、派斯瑞(Chord、Pastry)	卡萨(Kazza)
区块链应用	无(类似银行)	比特币	以太坊	超级账本
是否去中心化	否	是	是	否
是否可精确查找	否	否	是	否

2　比特币的 P2P 网络及节点发现

比特币开启了区块链时代，任何节点开启客户端后即可实现去中心化可信任的比特币交易。然而当一个全新的节点加入比特币网络时，首先要做的是接入网络。由于比特币的完全去中心化，节点自由的加入、退出，导致新加入的节点无从获取网络中节点地址从而接入网络。为此，比特币设计了如下三种节点发现方式。

1）种子节点

Napster 采用的是中央服务器作索引，由于中央服务器的存在，新加入节点可以稳定地接入网络。比特币虽然没有中心化服务器，但是也采用了 Napster 的思路，设立“种子”节点。比特币将一部分长期稳定的节点硬编码至代码中。这些节点在初始启动时，提供最初接入网络的入口节点。新节点通过这些稳定节点作为中介连接其他节点，并且可以持续获取区块链网络节点地址列表，所以这些节点也称之为种子节点。比特币源码[22]如图 3 所示，这些地址即为比特币初始化时加载的种子地址。

```
// Note that of those which support the service bits prefix, most only support a subset of
// possible options.
// This is fine at runtime as we'll fall back to using them as a oneshot if they don't support the
// service bits we want, but we should get them updated to support all service bits wanted by any
// release ASAP to avoid it where possible.
vSeeds.emplace_back("seed.bitcoin.sipa.be"); // Pieter Wuille, only supports x1, x5, x9, and xd
vSeeds.emplace_back("dnsseed.bluematt.me"); // Matt Corallo, only supports x9
vSeeds.emplace_back("dnsseed.bitcoin.dashjr.org"); // Luke Dashjr
vSeeds.emplace_back("seed.bitcoinstats.com"); // Christian Decker, supports x1 - xf
vSeeds.emplace_back("seed.bitcoin.jonasschnelli.ch"); // Jonas Schnelli, only supports x1, x5, x9, and xd
vSeeds.emplace_back("seed.btc.petertodd.org"); // Peter Todd, only supports x1, x5, x9, and xd
```

图 3　比特币的种子地址

2）地址广播

接入某个节点后，就可以以这个节点为中介，获取网络其他节点地址，这种方式被称为地址广播。广播包含如下两种方式：

a)主动广播(推)A 节点地址

比特币通过 Net. AdvertiseLocal( )方法，将自身节点信息推送给其他节点。如图 4 所示，这是一个主动单向过程，B 节点接收后只保存在本地，并不做回应。同样，B 节点与其他节点间，也可以通过推送自身地址传递给其他连接的节点，通过推的方式使得连接节点获得自身节点的地址信息。

b)主动获取(拉)其他节点地址

仅仅推送自身地址，只局限于自身连接的节点并不足以将地址广播出去，比特币还有另外一种地址广播机制，通过 Net.GetAddresses( )方法主动拉取地址。如图 4 所示，A 节点请求地址后获取 B 节点，B 节点返回 B 节点收录的地址信息。一次请求一次响应，以避免对网络资源的浪费。B 节点也可以通过向其他节点请求地址，以扩充自身地址库。其他节点主动推送的地址也在拉取动作时返回请求节点，达到地址广播的目的。比特币的两种地址获取方式是独立的，这样可以防止攻击者通过伪造大量假的地址并广泛散播，导致正常节点无法接入比特币网络或接入虚假网络。

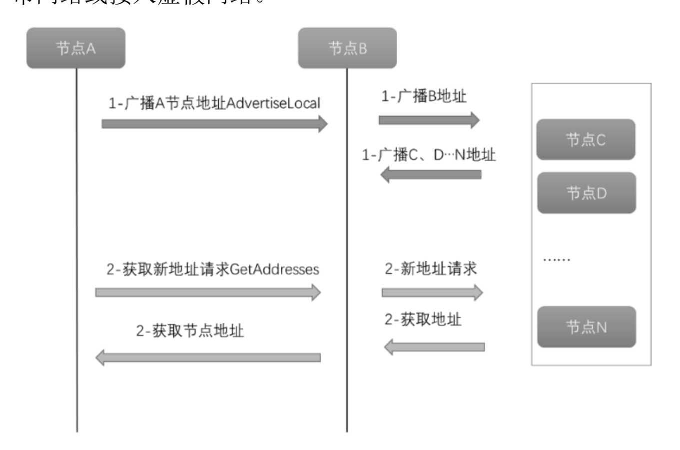


图 4　比特币地址广播流程

3）地址数据库

为避免种子节点连接数及带宽限制，比特币节点接入网络后通过广播获取其他节点信息，并将结果保存，以便下次使用时接入网络。比特币客户端将获取的节点信息保存在 peers.dat 本地文件中,使用的是 levelDB 格式储存数据。如果 A 节点与已

建立的连接 B 节点没有数据通信，A 节点会定期发送 ping 消息到 B 节点，一个 8 位的随机数。B 节点会回复 pong 消息，它是一个包含 ping 随机数的响应报文，以维持连接。如果节点持续某个连接长达 20 min 没有任何响应，它会被认为已经从网络中断开。比特币客户端启动后会发起定时循环任务，检查连接节点是否在线，并将失败的节点保存在 banlist.dat 本地文件中。

比特币通过以上三种方式保持稳定的网络接入，实现对频繁进出节点的地址列表维护，无须中心机构的存在即可保持节点可以自由加入网络，从而保障比特币 P2P 网络稳定。

3 以太坊的 P2P 网络及节点发现

以太坊是在比特币基础上发展而来，借鉴了很多比特币的思想，就连白皮书都有一半内容是描述比特币的功能。所以以太坊不仅能够实现类似比特币的交易系统，更希望构建基于区块链的生态环境，拓展依赖于以太坊衍生的分布式应用（decentralized application DAPP）。以太坊开发人员开发了 Whisper[23]一款分布式消息通信平台和 Swarm 一款分布式存储平台以及内容分发服务平台。这些应用的实现依赖于节点可以根据需要精确查找另一个节点地址的功能，而非结构化 P2P 网络结构无法满足。因此以太坊采用结构化 P2P 网络，通过 DHT 技术实现结构化。DHT 将 P2P 网络节点通过 Hash 算法散列为标准长度数据，整个网络构成一个巨大的散列表。每个参与节点都有一部分的散列表，并储存维护自身数据，散列表分布在 P2P 网络各个节点上。任何接入 P2P 网络的节点都有自身位于散列表中位置的 ID,可以通过 DHT 寻找更多节点，也可以被其他节点根据 ID 值精确查找。虽然 DHT 支持节点自由地加入或退出，但 DHT 的复杂维护机制，使其无法适应高频的节点变化。以太坊使用的是 Kadenlia [24](Kad)协议。Kad 是 DHT 协议的一种，使用该协议，可以快速准确地查找地址。

3.1 Kad 介绍

与传统的 DHT 比较， Kad 有如下优点：

a)Kad 的查询请求是并行、异步的，可以避免节点退出或故障导致的查询失败情况。

b)Kad 简化了节点间为了了解彼此而必须发送的配置消息数量，并且在查找时自动交换配置信息。

c)Kad 采用二叉树节点分为多个 Kad 桶，简化查询结构。使用单向性异或算法(XOR)计算距离，保证对同一个 TargetID 所有查询都会收敛到同一个路径，以减轻节点间查询网络消耗，提升查询效率。

d)节点在记录其他节点是否可用时所采用的算法可以阻止一些常见的拒绝服务(denial of service，DoS)攻击。

以太坊 Kad 与传统 Kad 相比，收敛方式不同。传统的 Kad 是以自身 selfID 作为收敛目标，而以太坊不仅以自身 selfID 为收敛目标，还通过随机生成的 TargetID 作为收敛目标，以生成较大范围的散列表。

3.2 以太坊节点发现

1）种子节点

a)硬编码种子节点

同比特币及其他 P2P 软件一样，新节点使用硬编码的种子节点，接入以太坊 P2P 网络。以太坊种子节点源码[25]结构如图 5 所示。它由节点信息、网络地址 URL 和数据协议三部分构成。节点信息是十六进制转换过的 128 bit 的节点 ID，网络地址 URL 是分割符@后的 IP 地址，数据协议是 TCP 监听端口号(listening port)30303 或者 UDP 发现端口号(discovery port)30301。

b)矿池种子节点

通过 IP 查询工具查询图 5 中地址，分别来自爱尔兰、美国、巴西、澳大利亚、新加坡以及德国。跨国通信的高延迟无法满足国内节点网络需求。为了更加便捷的接入网络，EthFans 发起星火节点计划。EthFans 鼓励国内对以太坊项目感兴趣的组织和个人分享其稳定运行节点地址信息，筛选其中优质节点，打包成 static-nodes.json 文件后发布。以太坊用户下载后可以连接到更多国内节点，加快以太坊国内网络速度。类似的配置文件还有 trusted-nodes.json 保存信任节点信息，这两个文件不限定内容，只限定文件名及格式，以固定文件名的形式硬编码保存在配置文件内，配置的节点会在以太坊启动时加载。静态节点信息如图 6 所示。

2）地址数据库

类似比特币，以太坊也采用地址数据库的形式保存历史连接过的节点。以太坊采用的是 leveldb 作为历史文件格式，生成多个*.ldb 历史数据库文件。对于首次连接以太坊网络的节点，地址数据库是空的，后续随着地址广播，逐渐填满地址列表。对于再次连接以太坊的节点，启动时以太坊使用 loadSeedNodes( )方法将种子节点和历史数据一起读取，快速高效连接以太坊 P2P 网络。

3）地址查询

以太坊 P2P 网络最大的特点是可以进行地址查询。以太坊的 Kad 参数配置如图 7 所示。

NodeID=节点标志符（节点椭圆加密后的公钥长度/8）=64 位长度数据，规范每个节点长度信息。

NBuckets= Kad 桶，共有 32*8/15=17 个，节点自身维护路由表的数量，K 桶越多，查找速度越快，消耗资源越多。

bucketMinDistance：最近距离是 256-17=239，这是 Kad 节点间的距离标准。

bucketSize：每个 Kad 桶包含 16 个节点，每个路由表包含节点数量；bucketSize 越大，查找速度越快，同样消耗越多资源。

refreshInterval 刷新 timer=30 * time.Minute，设定刷新频率。

revalidateInterval 验证 timer = 10 * time.Second，设定验证节点频率。

copyNodesInterval 持久化 timer = 30 * time.Second，设定持久化频率。

```
// MainnetBootnodes are the enode URLs of the P2P bootstrap nodes running on
// the main Ethereum network.
var MainnetBootnodes = []string{
    // Ethereum Foundation Go Bootnodes
    "enode://a979fb575495b8d6db44f750317d0f4622bf4c2aa3365d6af7c284339968eef29b69ad0dce72a4d8db5ebb4968de0e3bec910127f134779fbcb0cb6d3331163c@52.16.188.185:30303", // IE
    "enode://3f1d12044546b76342d59d4a05532c14b85aa669704bfe1f864fe079415aa2c02d743e03218e57a33fb94523adb54032871a6c51b2cc5514cb7c7e35b3ed0a99@13.93.211.84:30303",  // US-WEST
    "enode://78de8a0916848093c73790ead81d1928bec737d565119932b98c6b100d944b7a95e94f847f689fc723399d2e31129d182f7ef3863f2b4c820abbf3ab2722344d@191.235.84.50:30303", // BR
    "enode://158f8aab45f6d19c6cbf4a089c2670541a8da11978a2f90dbf6a502a4a3bab80d288afdbeb7ec0ef6d92de563767f3b1ea9e8e334ca711e9f8e2df5a0385e8e6@13.75.154.138:30303", // AU
    "enode://1118980bf48b0a3640bdba04e0fe78b1add18e1cd99bf22d53daac1fd9972ad650df52176e7c7d89d1114cfef2bc23a2959aa54998a46afcf7d91809f0855082@52.74.57.123:30303",  // SG

    // Ethereum Foundation C++ Bootnodes
    "enode://979b7fa28feeb35a4741660a16076f1943202cb72b6af70d327f053e248bab9ba81760f39d0701ef1d8f89cc1fbd2cacba0710a12cd5314d5e0c9021aa3637f9@5.1.83.226:30303", // DE
}
```

图 5 以太坊种子节点源码图

```
config.go ●   server.go   node.go node   service.go   doc.go   peer.go   node.go ...\discover

    "github.com/ethereum/go-ethereum/log"
    "github.com/ethereum/go-ethereum/p2p"
    "github.com/ethereum/go-ethereum/p2p/discover"
)

const (
    datadirPrivateKey      = "nodekey"            // Path within the datadir to the node's private key
    datadirDefaultKeyStore = "keystore"           // Path within the datadir to the keystore
    datadirStaticNodes     = "static-nodes.json"  // Path within the datadir to the static node list
    datadirTrustedNodes    = "trusted-nodes.json" // Path within the datadir to the trusted node list
    datadirNodeDatabase    = "nodes"              // Path within the datadir to store the node infos
)
```

```
static-nodes.json

[
 "enode://d8f4c028b96eeb53dcd87448962599cc14686d94e341e3d3ff51a9d313fa822bb434f5227c2b5f6b74da26fb82b291e79b23535a9d7e998701d21f1201c9287d@209.9.107.84:30303",
 "enode://d03335ec6b73fe55af464433361f6aa7b1aca92ad8cac24cbc76d88f16b39508c11b94de0ae9c6813dff7b18e3e3fcbb36124ab0273c92ba4c80eb32b2c0446c@222.210.114.171:30303",
 "enode://d5d953b27cd642436be70267157505230a9ff57423fe54d3249b804793514fb1d37dd48d3fa1abe7c01d53b1ffcf0f25b7ce0d0a62212b5e7e9565004b261ae4@222.210.114.171:30303",
 "enode://d3a88153add67753734154997da2dc61ce53a29565fbb0aaf22801e697c95000a7a625b760f60e0b87f959758380c3444a8d3712fa4a0767e1c4ea5252a34bed@211.83.110.142:30303",
 "enode://4b4e5013e0def80f893c7eaed4dab3ef92fa0d165db1a8ffa1bcaa4ac02d0bacc57d23ed10cbce5a54356a4e31bb8c171d1db1c76c40ace39a63acd1163d8bad@192.154.99.91:30303",
 "enode://ca33ab88e0db3965dc6faa77f9fd8d2a42b0f3ffbb8f87c4a45bb72103513681e428444b5079eed6e2b7cc8ae73b36165d0d0accba4b2c4f69523ffa9208f23e@123.57.234.11:30303",
 "enode://c7e3fc2774583e802357295d6937d904909c9318d2dbea41aaffa532030bf7fc001e13f9e951e0548e1b2b3df3039115c8f422a26a8c39ea3d7630ba6b7dd513@121.42.57.138:30303",
 "enode://4894cc9782242095b82d7097603738cdb964f515d0283f9a144a11cb87d7a3f019eaf9bd1370c19262426e1419191387476b531a217bf094bc5d052dd417790d@112.126.95.3:30303",
 "enode://12b3873357f59b4033d61ff03ec1eace647be8661daa2001a414984873c3db50433b203100a008c7888679accbdda89ebccff45d8b3ee82a0ed51a9791147e34@115.28.140.194:30303",
 "enode://c9ffd2250aea33e41ac63ca8d479ced251eaa8b1d66418e8219738fbcea50365fa3b3d958a49f2d3125b7712e318dba1c3775f2a886f44cf17f5193500ebf8cb@118.122.122.16:30303",
 "enode://d8909aa15b18eaaec9484b9a175545877300e257fb3acf17ad3b13b570d516753ca21a90ea690232896510f5fe2951aa6e3a2500ea130c616f70fe5cef9821c7@121.41.82.49:30303",
 "enode://2ecbd563b9a3a07d1836c6249a7f6d7f4599eda3e31ce120890a2916698081c94cf84823f528678bb28119bc0c51dd072c25209e735a00420fd78971e0c2d285@211.23.3.139:30303",
```

图 6 静态地址配置及内容

```
const (
    alpha           = 3  // Kademlia concurrency factor
    bucketSize      = 16 // Kademlia bucket size
    maxReplacements = 10 // Size of per-bucket replacement list

    // We keep buckets for the upper 1/15 of distances because
    // it's very unlikely we'll ever encounter a node that's closer.
    hashBits          = len(common.Hash{}) * 8
    nBuckets          = hashBits / 15       // Number of buckets
    bucketMinDistance = hashBits - nBuckets // Log distance of closest bucket

    // IP address limits.
    bucketIPLimit, bucketSubnet = 2, 24 // at most 2 addresses from the same /24
    tableIPLimit, tableSubnet   = 10, 24

    maxBondingPingPongs = 16 // Limit on the number of concurrent ping/pong interactions
    maxFindnodeFailures = 5  // Nodes exceeding this limit are dropped

    refreshInterval    = 30 * time.Minute
    revalidateInterval = 10 * time.Second
    copyNodesInterval  = 30 * time.Second
    seedMinTableTime   = 5 * time.Minute
    seedCount          = 30
    seedMaxAge         = 5 * 24 * time.Hour
)
```

图 7 以太坊科安达(Kad)配置参数

a)Kad 查找随机节点的步骤

(a)创建一个数据库实例 db 用于持久化储存信息。

(b)载入参数通过 newTable( )方法，生成 Kad 桶对象实例，初始化 K 桶数据。

(c)loadSeedNodes( )方法载入种子节点和旧节点数据，计算种子节点的 IDhash 值，与 bucketMinDistance 以比较选择 Kad 桶，写入数据。

(d)通过 GO 语言的 goroutine 运行库，一种类似并发线程概念，使 db 执行循环 loop( )方法

(e)loop( )方法做了三件事，首先重新加载种子节点，然后

查找随机生成三个节点并查找这三个节点的周围节点，最后将 Kad 桶节点信息保存进 db。

b)Kad 查找固定节点步骤

(a)先在当前的桶里面用 closest( )方法查找，获得最近于目标节点的临近节点。

(b)查找一遍之后没找到就在周边的节点中，使用 Lookup( )方法再搜索一遍。

(c)Lookup ( )会要求已知节点查找邻居节点，查找的邻居节点又递归的找它最近的 alpha（3）个节点 findnode()。

4）地址广播

由于有固定的 Kad 表结构，每个节点根据自身 NodeID 有唯一的固定位置，所以以太坊没有主动广播自身地址的方法。但是通过 doRefresh( )方法的 Lookup( )方法搜寻与自身最近的 16 个节点进行连接，接着随机生成三个 targetID，利用 Lookup( )方法寻找节点。寻找后利用 pingpong( )方法进行双向的通信，对方节点会将连接的节点比较远近后保存至自身 k 桶，同时把节点信息储存在地址数据库中，通过这种方式实现将自身节点地址告知其他节点的目标。

4 超级账本的 P2P 网络及节点发现

随着比特币以太坊的快速发展，区块链技术的应用范围不再局限于交易系统，对某些企业级问题，区块链也是完美的解决方案，如物流链、供应链等无中心系统下实现货物追踪，防窜改等问题。超级账本 Fabric 应运而生，Fabric 作为企业级区块链应用，各节点权限是不同的，交易处理必须经过超级节点才能完成。虽然 Fabric 没有实现去中心化，却可以通过划分不同节点之间的工作负载实现优化网络效率。为了保证区块链网络的安全性、可信赖性及可测量性，Fabric 采用 Gossip 作为 P2P 网络传播协议。Gossip 支持超级节点网络架构，超级节点具有稳定的网络服务和计算处理能力。超级节点负责 Fabric 的交易排序和新区块广播功能，维护 Fabric 网络信息更新、节点列表管理等内容。

4.1 Gossip 介绍

Gossip[26]是基于传染病感染传播机制而形成的，被广泛用于分布式系统作为底层通信协议。Gossip 并非新的 P2P 网络思想，同传统的洪泛、路由算法相比，Gossip 提供了明确的网络通信类型。Facebook 开发的 Cassandra[27]使用的就是 Gossip 协议，是一种流行的分布式结构化数据存储方案[28]。Gossip 在 Fabric 网络上负责维护新节点的发现、循环检查节点、剔除离线节点、更新节点列表。Fabric 通过与节点列表节点广播通信，发现新的区块或本地错误及丢失的区块，更新维护本地账本数据。Gossip 协议有如下三种通信类型：

a)push，主动信息推送。假设自身为 A 节点，随机选择节点列表中的 N 个节点作为目标节点。主动推送包含 A 节点自身 ID 的信息至 N 个节点，N 个节点接收后，对比自身数据，更新自身信息。N 个节点也可以继续 push 信息至其他节点。

b)pull，主动拉取信息。假设自身为 A 节点，随机选择节点列表中的 N 个节点作为目标节点。A 节点将自身 ID 和请求内容如节点列表、区块信息等封装成请求报文，主动发送给 N 个节点，N 个节点收到请求后，将相应信息返回给 A 节点，A 节点对比本地数据后更新自身本地数据。

c)push/pull，混合模式，虽然综合 push 与 pull 的优点，但是网络资源消耗较大。假设自身为 A 节点，pull 取 N 个节点信息后同 A 节点本地数据比较，再将较新数据 push 至 N 个节点，N 个节点比较后更新自身数据。

每个节点发送消息前，都会通过用自身节点的安全标志符 PKI_ID 和加密签名封装消息。恶意节点因为没有 Fabric 证书颁发机构 (certificate authority，CA)认证的密钥，从而无法冒充其他节点发送信息，保障通信安全性。Gossip 的启动流程的 fabirc 源码[29]如图 8 所示。

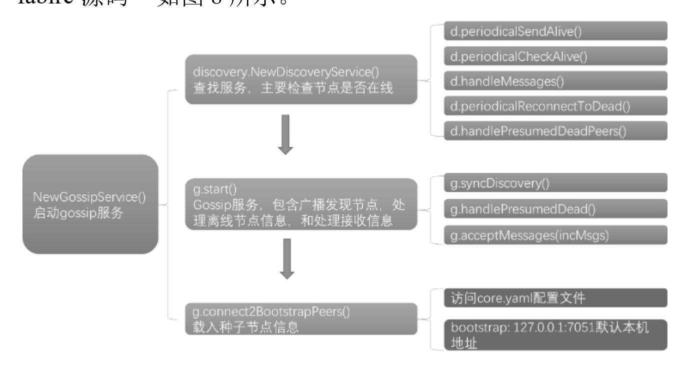


图 8 绯闻协议(Gossip)启动流程

4.2 Fabric 节点发现

1）种子节点

Fabric 采用的不是硬编码的形式，而是配置文件的形式通过 core.yaml 配置文件进行配置。以太坊采用 viper[30] 作为配置文件加载方案。viper 是国外计算机爱好者昵称 spf13 编写的开源配置解决方案。viper 不用关心文件格式,可以获取本地环境变量,也可以从远端获取配置文件,同时还有缓冲机制，可以在不重启服务的情况下动态加载新的配置项的值并使之实时生效。这对 Fabric 网络至关重要，超级节点管理员可以在不影响系统运行，无须重启系统的情况下，更新配置信息并使之生效。而比特币和以太坊由于去中心化，需要变更只能进行全部节点投票，一旦无法达成一致则造成分叉，如比特币分叉为比特币(BTC)和比特币现金(BCC), 以太坊则分叉为（ETH）和以太坊经典（ETC）。种子节点相关源码如图 9 所示。种子节点的加载流程如下：

a)首先启动 Gossip 服务 NewGossipService( )方法，在服务启动过程中调用 g.connect2BootstrapPeers( )方法加载种子节点。

b)通过 g.conf.BootstrapPeers( )方法，读取 core.yaml 配置文件，获取 bootstrap 超级节点的值，保存为 endpoint 终端地址临时变量。

c)随后启动连接 g.disc.Connect( )方法，将 endpoint 作为参数传入。

d)使用 d.createMembershipRequest( )方法生成请求信息，并

赋值给临时变量 m 并加密签名为请求信息 req。

e)将 endpoint 和自身 PKIid 组合，利用 go d.sendUntilAcked( ) 方法将 req 信息发送至对应 endpoint。

f)获得 endpoint 节点返回信息后更新地址库。

```
NewGossipService ()

g.chanState = newChannelState(g)
g.emitter = newBatchingEmitter(conf.PropagateIterations,
    conf.MaxPropagationBurstSize, conf.MaxPropagationBurstLatency,
    g.sendGossipBatch)

g.discAdapter = g.newDiscoveryAdapter()
g.disSecAdap = g.newDiscoverySecurityAdapter()
g.disc = discovery.NewDiscoveryService(g.selfNetworkMember(), g.discAdapter, g.disSecAdap, g.disclosurePolicy)
g.logger.Info("Creating gossip service with self membership of", g.selfNetworkMember())

g.certPuller = g.createCertStorePuller()
g.certStore = newCertStore(g.certPuller, g.idMapper, selfIdentity, mcs)

if g.conf.ExternalEndpoint == "" {
    g.logger.Warning("External endpoint is empty, peer will not be accessible outside of its organization")
}

go g.start()
go g.connect2BootstrapPeers()

return g
}
```

```
core.yaml | gossip_impl.go | discovery.go | discovery_impl.go

# Gossip related configuration
gossip:
    # Bootstrap set to initialize gossip with.
    # This is a list of other peers that this peer reaches out to at startup.
    # Important: The endpoints here have to be endpoints of peers in the same
    # organization, because the peer would refuse connecting to these endpoints
    # unless they are in the same organization as the peer.
    bootstrap: 127.0.0.1:7051
```

图 9　种子地址加载及配置图

2）地址数据库

Fabric 采用超级节点的形式，都需要接入超级节点。而且使用场景是企业级应用，所以无须保存历史数据。因为保存数据、验证节点是否在线需要耗费大量资源。既然有超级节点的存在，只需要每次去超级节点取地址即可。超级节点也完成了地址数据库的工作。加载种子地址后，将获取节点列表保存至内存中，而非比特币和以太坊的文件形式。节点通过解析收到消息，检查节点是否正常，维护节点列表，不仅如此还定时与连接节点通信，一旦被连接节点超过配置时间没有响应，则将其移出节点列表，加入离线列表。

3）地址广播

Gossip 通过启动 g.syncDiscovery( )这一个循环方法，定时 S 秒循环进行查找节点工作。通过在节点列表中随机选择 N 个节点，push 索要节点列表信息，同步自身节点列表。core.yaml 中默认配置 S 为 4 s，N 为 3 个节点，可以根据需求进行更改。在 push 的同时也将自身节点信息传递给其他节点，并通过广播传递至整个网络。

4）地址查询

Fabric 没有业务场景需要精确查询某个 ID 的地址，所以目前不支持地址查询。但是因为超级节点的存在，超级节点拥有所有已连接节点地址信息，可以通过扩展 Fabric 代码，增加功能的方式实现查找。

5　分析对比

每种 P2P 网络结构拥有各自的优缺点，针对不同区块链应用场景、不同的业务需求，P2P 网络的使用也不同。区块链主要应用场景是代币交易系统。为了更直观地对比不同区块链系统 P2P 网络结构，使用银行类中介系统替代中心化网络结构的产品。网络结构属于架构设计层次，与功能点不同，故本文只描述不同区块链产品网络结构组成，不评分对比。区块链作为创新产品其主要特点是去中心化、匿名信、可信任及智能合约，这些是功能上的创新。本文通过分析区块链产品源代码的技术实现，分析比较其底层架构及技术特点，对应着从去中心化程度、隐私保护、安全性、应用的丰富程度四个维度进行分析评价，每个维度代表不同功能点，功能点属性具有相同的权重。P2P 网络虽然不是区块链的创新特点，但其作为区块链底层技术支撑，网络质量决定着区块链产品的成败。本文从源代码技术角度分析，将 P2P 网络作为区块链的一个功能点进行分析对比，通过从接入效率维度评价区块链 P2P 网络优劣，从而合理假设五个评价维度皆为功能点属性，权重相同。

1）去中心化程度

P2P 网络是区块链分布式的基石，自从比特币成功以来，区块链分布式去中心化概念深入人心。传统中心金融机构所有交易必须经中央系统处理，参与节点间无法直接通信，是完全的中心化。比特币和以太坊都是完全的去中心化，所有节点权限相同，虽然有种子节点的存在，但架构上和普通节点没有区别，只是相对高性能、高稳定性的节点，在去中心化上两者表现相同。Fabric 则相当于分布式的中心化系统，超级节点具有中心化服务器功能，同时超级节点也是分布式集群，具有分布式特征，所以 Fabric 去中心化表现介于银行类系统和比特币以太坊之间。

2）节点接入网络效率

节点接入网络需要做两件事，一是需发现节点接入区块链网络；二是需同步区块账本数据等信息，才能使用区块链服务。银行类中心系统所有维护工作都由中心系统负责，节点接入即可使用服务，所以节点可以快速接入退出，接入效率最高。由于没有中心节点，账本保存在本地，比特币和以太坊都需要同步全部区块后才进行共识挖矿。但比特币节点地址管理无须构建维护 DHT，可以直接广播，无须进行查找，所以节点发现效率上高于以太坊。Fabric 有可以灵活配置的超级节点，所以接入效率高于比特币，但弱于有强大性能服务器的银行类金融系统。

3）安全性

常见的区块链攻击方式有两种，一种是节点伪造，冒充其他节点进行交易；另外一种是 DOS 攻击，攻击服务提供者使系统瘫痪。中心化金融系统除了采用多种加密方式外，还绑定个人信息，在交易过程中结合人脸识别[31]、PIN 密码[32]、指纹及绑定个人信息的智能卡[33]、与密码结合的双重保障[34]等多种方式确保交易安全，强大的网络及服务器能力也使 DOS 攻击变得非常困难。比特币与以太坊采用账户与节点分离的设计，不限制节点的加入与退出，使得节点伪造失去意义；所有节点通过共识机制提供服务，无中心系统，DOS 只能攻击整个网络，而这样代价是巨大的，迄今为止没有成功攻击比特币与以太网的 DOS 的案例。Fabric 采用 CA 节点分配秘钥，避免伪造节点，同时利用分布式超级节点抵抗攻击。Fabric 在安全性上相对比

特币以太坊增加节点管理，但认证要求弱于银行类系统，所以安全性介于两者之间。

4）隐私保护

身份信息隐私保护分为两个层面：基本层次是身份标志保护；高级层次是用户登录行为的不追踪性[35]。由于区块链去中心化特点，隐私问题在区块链中对应为基本层次节点 IP 地址是否匿名，高级层次身份信息是否保密两个方面。节点 IP 地址包含物理地理信息等，可以用做节点的身份标志。但存在一个物理节点多个用户使用情况，而身份信息则可以精确定位用户，追踪用户行为。以太坊由于有 DHT 保存节点 IP 地址信息，有可能被针对性地攻击。账户方面同比特币一样，不包含身份信息以保护隐私。Fabric 节点信息保存在中心服务器。虽然节点间采用随机连接的方式，但 IP 地址、身份信息等内容全部暴露给超级节点，对于超级节点是公开的，对于普通节点是匿名的，所以匿名性介于中心系统与分布系统之间。对银行类中心系统，身份信息是认证信息的一部分。身份证等信息对中心服务器是公开的，但节点间无法通信。银行类中心系统负责保护身份信息，一旦中心系统作恶，则有可能泄露隐私信息。比特币采用洪泛的方式广播，无法定位节点是信息最初发出节点还是转发节点；保护节点地址信息；账户采用匿名制，保护用户身份信息。

5）应用的丰富程度

P2P 网络使得区块链系统内节点间可以互相通信，构成系统网络通信底层，在此基础上区块链上层延伸出了一系列应用。银行类系统虽然功能丰富，但因为是中心化系统，参与者无法发布智能合约、分布式扩展应用等，限制了应用的扩展。比特币由于不支持智能合约，主要功能依旧是交易系统，所以扩展应用最少。以太坊 DHT 支持精确查找，可以精确定位节点或范围内节点地址进行通信，如点对点通信、点对点文件传输等功能，在应用丰富程度上表现最佳。Fabric 支持智能合约，可以像以太坊一样构建扩展应用，但由于没有 DHT，不支持精确地址点对点通信，应用扩展功能上弱于以太坊。

为了量化评价不同 P2P 网络结构在区块链中的表现，按照表现优劣使用 1~4 分对不同产品进行评分，分数越高表现越优秀。评分结果如表 2 所示。

表 2　网络协议评分表

评分项	银行类金融中介系统	比特币	以太坊	超级账本
网络结构	中心化网络结构	全分布式非结构化网络结构	全分布式结构化网络结构	半分布式网络结构
去中心化程度	1	4	4	2
节点接入网络效率	4	2	1	3
安全性	4	2	2	3
隐私保护	2	4	3	3
应用的丰富程度	3	1	4	3
合计	14	13	14	14

本文对目前主流区块链产品的 P2P 网络结构进行了分析，P2P 网络有效地解决了区块链系统节点数据一致性的问题，是区块链分布式账本得以实现的基石。然而每种网络结构各有特点，不同的区块链的应用场景使用不同的 P2P 网络结构。从表 2 数据单项比较，优劣可以明显区分，但从合计分数看，不同产品总分差距不大。仅凭合计数据无法表现区块链整体形成，综合本文对比的五个方面，按得分作出雷达图，可以看到每个产品整体的能力分布和侧重点，如图 10 所示。

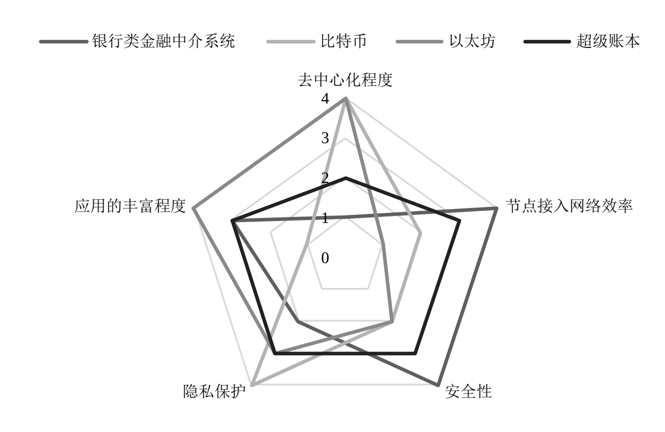


图 10　区块链产品对比雷达图

银行类系统网络虽然并非 P2P，但作为典型的中心化网络，是与分布式区块链产品对比的标杆。其高效的接入效率及安全性保障了系统的稳定性。虽然牺牲了分布式和扩展性，但作为中心系统，稳定高效胜于一切。比特币 2008 年发布以来，对于整体架构没有多大的变更，功能上也只是满足交易需求。其创始人中本聪也在比特币平稳运行后销声匿迹。比特币以其去中心化和匿名隐私保护性为特点，奠定了区块链的基础。虽然比特币总体只有 13 分，但隐私保护和去中心化都表现最好，哪怕是区块链快速发展十年后的今天，也是可圈可点。以太坊试图建立基于区块链底层的生态圈，创立了分布式应用 DAPP 的概念，其应用丰富程度是所有区块中最高的，代表应用以太猫 CryptoKitties 中价格最高的一只猫，以 246.95 个以太币的价格成交，约为 11.8 万美金、77 万人民币。超级账本作为一种企业级区块链应用，采用联盟链的形式，介于中心化与非中心化之间，在图 10 中呈五边形状，每个维度表现比较均衡，没有明显的优势同时也没有明显劣势。

6　结束语

区块链技术构建了可信任去中心化分布式应用体系，去除完全的中心化系统，降低社会价值交换成本，具有广泛的应用前景。区块链技术也不再局限于比特币单纯的交易系统，延伸至以太坊的基于区块链为底层的扩展应用；不局限于比特币的完全去中心化，延伸至 Fabric 的半分布式网络以解决联盟式商业问题。为了适应不同应用场景的需求，区块链的 P2P 网络协议也在持续演进。本文对区块链 P2P 网络的演进过程从底层源代码及上层应用需求角度进行梳理和归纳，以期为未来研究提供有益的启发与借鉴。未来随着区块链技术的发展，应用场景

逐渐丰富，区块链网络协议也会逐渐演进，这会是一个持续的演进过程。

参考文献：

[1] 罗杰文. Peer-to-Peer 综述 [EB/OL]. [2018-07-06]. http://www. intsci. ac. cn/users/luojw/P2P/. (Luo Jiewen. The summarize of Peer-to-Peer [EB/OL]. [2018-07-06]. http://www. intsci. ac. cn/users/luojw/P2P/.)

[2] 周文莉, 吴晓非. P2P 技术综述 [J]. 计算机工程与设计, 2006, 27 (1): 76-79. (Zhou Wenli, Wu Xiaofei. Survey of P2P technologies [J]. Computer Engineering and Design, 2006, 27 (1): 76-79.)

[3] 王学龙, 张璟. P2P 关键技术研究综述 [J]. 计算机应用研究, 2010, 27 (3): 801-805, 823. (Wang Xuelong, Zhang Jing. Survey on peer-to-peer key technologies [J]. Application Research of Computers, 2010, 27 (3): 801-805, 823.)

[4] 刘小敏, 史银龙. 基于 P2P 网络的信息检索 [J]. 北京工业职业技术学院学报, 2009, 8 (4): 17-19. (Liu Xiaomin, Shi Yinlong. Information retrieval based on peer to peer network [J]. Journal of Beijing Polytechnic College, 2009, 8 (4): 17-19.)

[5] 谭庆丰, 方滨兴, 时金桥, 等. StegoP2P: 一种基于 P2P 网络的隐蔽通信方法 [J]. 计算机研究与发展, 2014, 51 (8): 1695-1703. (Tan Qingfeng, Fang Binxing, Shi Jinqiao, *et al.* StegoP2P: a hidden communication approach in P2P networks [J]. Journal of Computer Research and Development, 2014, 51 (8): 1695-1703.)

[6] 邵奇峰, 金澈清, 张召, 等. 区块链技术: 架构及进展 [J]. 计算机学报, 2018, 41 (05): 969-988. (Shao Qifeng, Jin Cheqing, Zhang Zhao, *et al.* Blockchain: architecture and research progress [J]. Chinese Journal of Computers, 2018, 41 (5): 969-988.)

[7] 刘敖迪, 杜学绘, 王娜, 等. 区块链技术及其在信息安全领域的研究进展 [J]. 软件学报, 2018, 29 (7): 2092-2115. (Liu Aodi, Du Xuehui, Wang Na, *et al.* Research progress of blockchain technology and its application in information security [J]. Journal of Software, 2018, 29 (7): 2092-2115.)

[8] 高峰, 毛洪亮, 吴震, 等. 轻量级比特币交易溯源机制 [J]. 计算机学报, 2018, 41 (5): 989-1004. (Gao Feng, Mao Hongliang, Wu Zhen, *et al.* Lightweight trans tracing technology for bitcoin [J]. Chinese Journal of Computers, 2018, 41 (5): 989-1004.)

[9] 叶聪聪, 李国强, 蔡鸿明, 等. 区块链的安全检测模型 [J]. 软件学报, 2018, 29 (05): 1348-1359. (Ye Congcong, Li Guoqiang, Cai Hongming, *et al.* Security detection model of blockchain [J]. Journal of Software, 2018, 29 (5): 1348-1359.)

[10] Fukumitsu M, Hasegawa S, Iwazaki J, *et al.* A proposal of a secure P2P-type storage scheme by using the secret sharing and the blockchain [C]// Proc of International Conference on Advanced Information Networking and Applications. Taipei: IEEE Press, 2017: 803-810.

[11] Gattermayer J, Tvrdik P. Blockchain-based multi-level scoring system for P2P clusters [C]// Proc of International Conference on Parallel Processing Workshops. Bristol: IEEE Press, 2017: 301-308.

[12] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system [EB/OL]. (2008-10-31) [2018-07-06]. http://www. bitcoin. org/bitcoin. pdf.

[13] Buterin V. Ethereum white paper [EB/OL]. [2018-07-06]. https://github. com/ethereum/wiki/wiki/White-Paper.

[14] Buterin V. Ethereum homestead documentation [EB/OL]. [2018-07-06]. http://ethdocs. org/en/latest/network/connecting-to-the-network. html#the-ethereum-network.

[15] Androulaki E, Manevich Y, Muralidharan S, *et al.* Hyperledger fabric: a distributed operating system for permissioned blockchains [C]// Proc of the 13th EuroSys Conference. New York: ACM Press, 2018: 1-15.

[16] Garnett N. Digital rights management, copyright, and Napster [J]. ACM SIGecom Exch, 2001, 2 (2): 1-5.

[17] Lyu Qin. Search and replication in unstructured peer-to-peer networks [C]// Proc of the 16th International Conference on Supercomputing. New York: ACM Press, 2002: 84-95.

[18] Jain S, Mahajan R, Wetherall D. A study of the performance potential of DHT-based overlays [C]// Proc of Usenix Symposium on Internet Technologies and Systems. Berkeley: USENIX Association Press, 2003: 141-154.

[19] Stoica I, Morris R, Karger D, *et al.* Chord: A scalable peer-to-peer lookup service for internet applications [C]// Proc of Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications. New York: ACM Press, 2001: 149-160.

[20] Rowstron A I T, Druschel P. Pastry: scalable, decentralized object Location, and routing for large-scale peer-to-peer systems [C]// Proc of IFIP//ACM International Conference on Distributed Systems Platforms. Heidelberg: Springer-Verlag Press, 2001: 329-350.

[21] 蒋成. 混合式对等网络 Kazaa 模型结构及其分析研究 [J]. 信息安全与技术, 2013, 4 (6): 69-71. (Jiang Cheng. Research on major model of peer-to-peer network [J]. Information Security and Technology, 2013, 4 (6): 69-71.)

[22] Nakamoto S. Bitcoin source code [EB/OL]. [2018-08-19]. https://github. com/bitcoin/bitcoin

[23] James Ray. Whisper official document [EB/OL]. [2018-06-27]. https://github. com/ethereum/wiki/wiki/Whisper.

[24] Maymounkov, Petar, Mazi, *et al.* Kademlia: a peer-to-peer information system based on the XOR metric [C]// Proc of the 1st International Workshop on Peer-to-Peer Systems. London: Springer-Verlag, 2002: 53-65.

[25] Buterin V. Ethereum go-ethereum source code [EB/OL]. [2018-08-19]. https://github. com/ethereum/go-ethereum.

[26] Demers A, Greene D, Houser C, *et al.* Epidemic algorithms for replicated database maintenance [C]// Proc of the 6th Annual ACM Symposium on Principles of Distributed Computing. New York: ACM Press, 1987: 1-12.

[27] Lakshman A, Malik P. Cassandra: a decentralized structured storage system [J]. ACM Sigops Operating Systems Review, 2010, 44 (2): 35-40.

[28] 李励. 分散的结构化存储系统——Cassandra [J]. 东方企业文化, 2013

(15): 142-143. (Li Li. Decentralized structured storage systems-Cassandra [J]. Oriental Enterprise Culture, 2013 (15): 142-143.)

[29] Foundation L. Hyperledger fabric source code [EB/OL]. [2018-08-19]. https://github. com/hyperledger/fabric.

[30] Spf13. Viper: go configuration with fangs [EB/OL]. [2018-07-06]. https://github. com/spf13/viper.

[31] 刘筱攸. 招行首推刷脸取款"人脸识别"再下一城 [N]. 证券时报, 2015-10-15 (A06) . (Liu Xiaoyou. The cmb bank to brush face withdraw money"face recognition"next city [N]. The securities times, 2015-10-15 (A06) .)

[32] Wang Ding, Gu Qianchen, Huang Xinyi, *et al.* Understanding human-chosen PINs: characteristics, distribution and security [C]// Proc of ACM on Asia Conference on Computer and Communications Security. New York: ACM Press, 2017: 372-385

[33] Wang Ding, Wang Ping. Two birds with one stone: two-factor authentication with security beyond conventional bound [J]. IEEE Trans on Dependable and Secure Computing, 2018, 54 (4): 708-722.

[34] Krol K, Philippou E, Cristofaro E D, *et al.* "They brought in the horrible key ring thing!"analysing the usability of two-factor authentication in UK online banking [EB/OL]. [2018-08-19]. https://www.researchgate.net/publication/271140632_They_brought_in_the_horrible_key_ring_thing_Analysing_the_Usability_of_Two-Factor_Authentication_in_UK_Online_Banking.

[35] Wang Ding, Wang Ping. On the anonymity of two-factor authentication schemes for wireless sensor networks: attacks, principle and solutions [J]. Computer Networks, 2014, 73 (C): 41-57.